Wireless IC Node with Compression Heuristics (W.I.N.C.H.)

Steve Jocke, Kyle Ringgenberg, and Stuart Wooters

***Abstract*—It is a well established fact within the world of Body Area Networks that the power consumption needed to maintain wireless connectivity dominates that of the rest of the system. This paper proposes a three-fold approach to reduce this transmission power, and thus, reduce the total power consumption of any arbitrary wireless node; in this case, an electrocardiogram monitor. The methodology addressed includes data compression, sub-threshold processor operation, and high-speed wireless bursting.**

***Index Terms*—University of Virginia, ECE 686, Body Area Networks, Embedded Compression, Sub-Threshold PIC, GHz Mixer.**

I. Introduction

POWER consumption has become the primary limiting factor for the vast majority of mobile networks, including body area networks (BANs). There has been a large quantity of work developed to address the energy concerns of these networks. This work can be classified into two categories of energy reduction: the node and the channel. Approaches to the node include, among others, minimizing power supply voltage and implementing power gating techniques. While these techniques have been widely successful, it is a well established fact that the energy needed to wirelessly broadcast the data from each node significantly outweighs nearly all on-chip demands. Approaches to this problem are generally limited to the development of new transmission protocols such as ZigBee, Wireless USB, and Wibree. [1] These protocols are a good start, but certainly do not constitute an all encompassing solution.

This work represents the design of three tiered system aimed at minimizing power consumption. The first measure taken to limit power usage is the implementation of a real time compression algorithm. By limiting the total number of bits that must be pushed over a channel, one can decrease the total period of time a system's wireless device must remain active. Several algorithms were developed and analyzed according to the metrics of compression ratio, signal fidelity, and implementation complexity. The second measure is the design and implementation of a sub-threshold processing core. By lowering the supply voltage of a wireless system's CPU, one can drastically decrease the power that unit consumes. This core was evaluated for operation as well as the availability of spare instruction cycles to implement the aforementioned compression. The final approach is the development of a high-speed wireless mixer. By increasing the wireless bandwidth, one may reduce the period of time a wireless element must remain active and may put said item into a deep sleep mode in between data bursts.

II. Compression

While there has been significant previous work regarding the implementation of compression algorithms in ASICs to minimize data I/O [2], this work provides room for extension in multiple realms. The work addressed here addresses the exploration of two of these realms. First is the matter of application; as previously addressed, for most body area networks, the energy consumption needed for transmission greatly dominates that needed for on chip calculation. This realization provides the motive to compress outgoing data, not for bandwidth concerns, but for power management. The fewer total bits projected across the channel is directly related to the total power that must be consumed. In this way, the total power expenditure of a given design may be decreased with limited to no loss in signal fidelity.

The second extension on this previous work relates to the fact that it does not provide any degree of expandability for variable applications. Compression ratios can be highly dependent on input data, so an algorithm developed for one use may be entirely inappropriate for another. If a designer is willing to sacrifice real estate and a degree of energy efficiency, he could implement a dedicated system core to perform reprogrammable compression algorithms prior to transmission. In this way, a single ASIC could be utilized for a wide range of applications while still providing the data compression, and thus, minimized transmission bit count.

As a means to remain cohesive with the whole of the system, three electrocardiogram (ECG) signals have been chosen to benchmark all compression variants. The first, ECG1, is a simplified idealization which helps verify functionality, but provides no real data. The second, ECG2, is sampled from a healthy individual at rest (1). The third, ECG3, is sampled from a healthy individual moving from rest into motion (2). Two general compression schemes have been chosen to be applied to these sample ECG signals: run length encoding and delta encoding. Each algorithm is swept over a variety of parameters to determine an optimal configuration. The two configurations with the greatest compression ratios (one lossey, one lossless) have been implemented in the PIC assembly language and tested for power consumption overhead as well as general performance.

A. Run Length Encoding

One method of data compression, run length encoding (RLE), involves the removal of multiple occurrences of like values from a data stream. At a high level, if a data stream consists of the following values: [4 9 9 9 9 9 8 7], one could

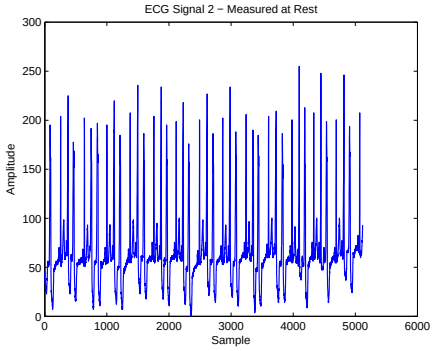

Fig. 1. Sampled ECG2 Waveform

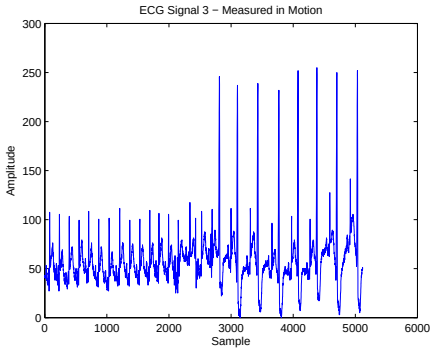

Fig. 2. Sampled ECG3 Waveform

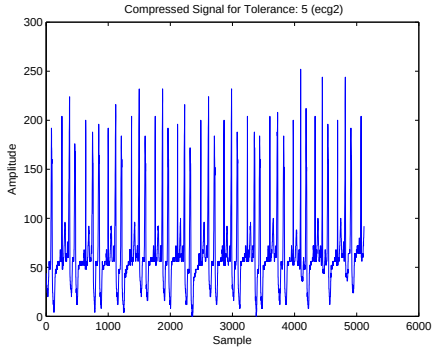

Fig. 3. Compressed ECG2 Waveform at 0.05 Tolerance

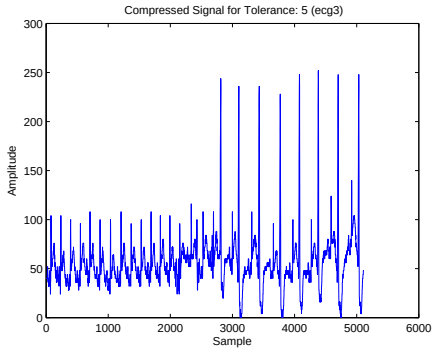

Fig. 4. Compressed ECG3 Waveform at 0.05 Tolerance

reduce the string of 9s to a single 9 with a repetition value. Thus, the 8 element stream is recoded as [4 (5)9 8 7]. This method provides lossless compression as the original stream may be entirely reconstructed from the compressed stream. By allowing a tolerance, such as accepting that the values 7 and 8 are similar enough to 9 to be considered equal, this example data stream may be compressed even further to: [4 (7)9]. This variant is a lossey scheme as the compressed stream actually resents a loss in data values. This loss in fidelity is akin to low-pass filtering the original signal.

In an actual system, the repetition values of an RLE stream must be delineated from the data values. By adding an additional codeword to a data stream every N words, one can express this additional information with a very limited impact on runtime performance. Considering that all words in the system implemented herein are 8-bits in length, one can express 4 variants on codeword implementation. If 1 bit is dedicated to repetition values, the longest strings that may be represented are 2 words. Likewise, 2 bits yield 4 words, 4 bits yield 16 words, and 8 bit yield 256 words. Furthermore, a new codeword must be inserted every 8 / bit output words. This presents an obvious trade off: the more bits per codeword allotted for repetition values, the longer the data stream repetitions may be, but the more often a codeword must be inserted. As a bit-level example, for 2-bit codes, the data stream: [FF FF FF 05 05 05 05 C2 C2 3D] can be losslessly encoded as: [B4 FF 05 C2 3D] which yields a 2x compression ratio. Figures (5) and (6) illustrate the effectiveness of variable codewords on the two benchmark ECG signals. Note that, for tolerance values above 0.05 (5% variability from maximum), a codewords of size of 8 dominates. Furthermore, 0.05 tolerance yields nearly unnoticable signal loss as evidenced in figures (3) and (4). Thus, for the target ECG signals, RLE privides supurb compression ratios with minimal signal degredation.

B. Delta Encoding

Delta encoding is based on sending difference values between samples rather than the samples themselves. The assumption is that the change between two points is greater in magnitude than the points themselves and for smooth waveforms, this assumption holds. A glance at the histograms for the derivatives of the two ECG benchmark signals (7) and (8) shows that one would expect the number of bits required to encode the delta values to be far less than the 8 per point needed to represent the raw values. To exploit this density of low valued deltas, three compression schemes were analyzed; one lossey and two lossless.

The most straightforward method of delta encoding is simply passing the delta values rather than the original point values. This in and of itself produces no gain in compression ratio, however, if the delta values are represented in a fewer number of bits, a gain is obtained. The lossey method of delta compression involves setting an upper limit on delta values based on the mean delta. For example, dictate that all delta values shall reside within [0, 7] yielding a delta bit size of 3. If a delta is encountered that is greater than the upper limit, it is simply replaced with the highest possible. In this way, only 3 delta bits are used to represent the original 8 bits

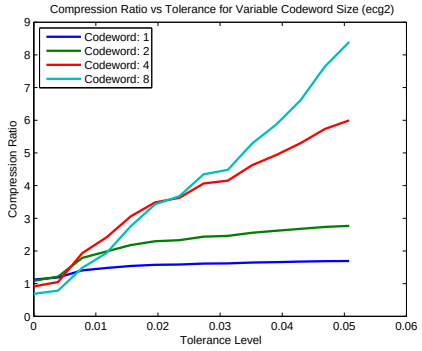

Fig. 5. Tolerance vs Compression Ratio for Variable Code Size (ECG2)

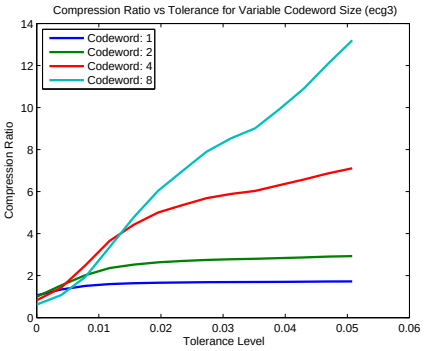

Fig. 6. Tolerance vs Compression Ratio for Variable Code Size (ECG3)

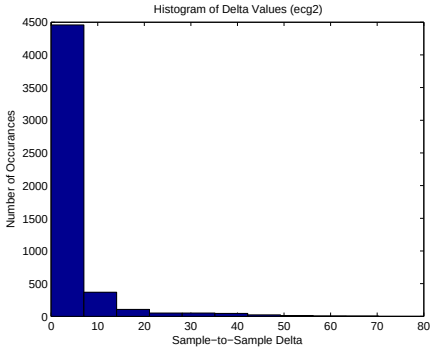

Fig. 7. Histogram of the Delta Values of ECG2

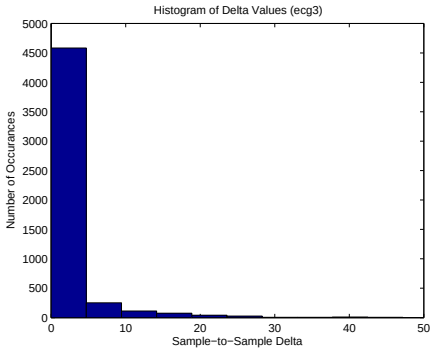

Fig. 8. Histogram of the Delta Values of ECG3

per word. This scheme shall be referred to as deltaG as it makes use of a globally defined delta maximum. This same procedure can be modified to a lossless compression scheme determining the worst case delta size and encoding all values accordingly. Thus, if it can be guaranteed that no two adjacent points will be greater than 64 values apart, each delta could be encoded as 6 bits, while still preserving the original waveform. This scheme shall be refered to as deltaG' as it too utilizes a global maximum. The limitation of this lossless approach is, of course, that if a delta occurs out of bounds, it suddenly becomes lossey and must resort to replacing the unusually large delta with a the maximum possible.

The final variant of delta coding that was investigated involves sending only the number of bits required to represent each delta value on a word-by-word basis. It shall be referred to as deltaL, referencing the local nature of this scheme. In this way, the original signal is able to be entirely reconstructed from the compressed signal, regardless of extreme jumps in delta values. For each delta value, a set length codeword is added to represent how many bits compose said delta value. For 8 bit word values the worst case delta jump can be dictated with 3 bits (assuming code 000 references 1 bit). Thus, each delta is represented with the minimum number of bits albeit with a reasonably significant overhead. Figure (9) shows the attainable compression ratios for the three delta encoding schemes described.

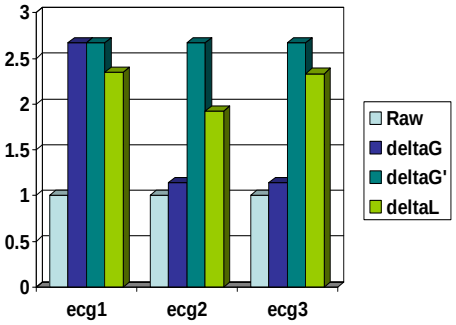

Fig. 9. Compression Ratios for Delta Encoding

C. Instruction Set Implementation

Of all the compression variations reference thus far, two were chosen for implementation in the PIC16 assembly language. Since RLE provides excellent compression ratios within its lossey variation, it has been implemented with a 0.05 tolerance level. After optimization, this compression algorithm was able to be fully implemented with 50 instructions and 7 registers. During execution, it utilizes approximately 30 instruction clock cycles and yields an average compression ratio of 10 to 1. Delta encoding using the localized scheme was implemented for its good lossless performance. This algorithm requires 44 instructions and 6 registers. During execution, it also uses approximately 30 instruction cycles while yielding an average compression ratio of just over 2 to 1.

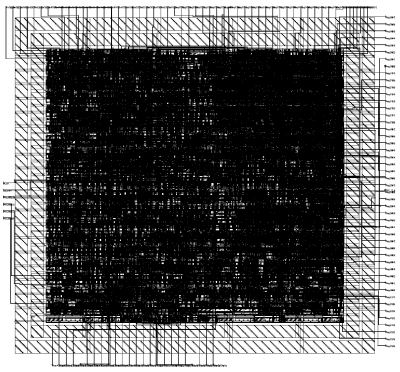

Fig. 10. PICCPU layout.

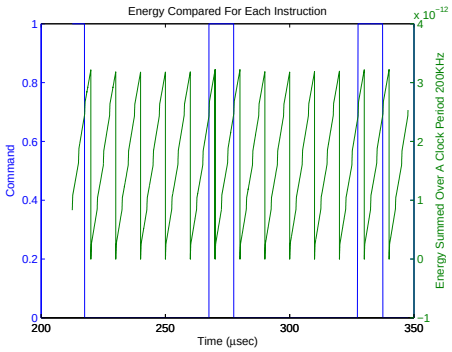

Fig. 12. PICCPU power consumption per cycle.

III. PICCPU

For this system an open source core of a PIC16C5X.UVa (10) shall be used. This core is based on the PIC16C5X from opencore.org. While most of the features of the PIC16C5X remain we have added a few extras to help suit the heart rate application. These two features are interrupts and more output ports. The four new interrupts allow for better interaction between a few custom blocks which removes the need for port polling. This will help to reduce code complexity and free up available instruction cycles (i-cycles). The 5 new output ports are used to alleviate communications issues between components. This design was chosen over a bus system due to a lower complexity and the abundance of available ports. The full system diagram can be seen below in Figure (11).

In this paper everything except the PIC processor shall be removed or replaced by a VHDL test bench. The general flow of data starts from the ADC unit, then to the PIC for processing, then to the UART. This system is designed to replace a prototype which currently uses a TI MSP430 Microprocessor.

A. System Evaluation

To decide if the system needs to be modified for implementation of compression algorithms a few checks are required. The first being original MSP430 code needs to be translated to PIC code. Having this information will reveal how many free i-cycles are remaining for the two compression algorithms. The PIC has the responsibility of doing gain control for the ADC. In order to complete this task there are two sections of code. The first keeps track of the highest value of a period of time. The second section of code takes the value and identifies what range it sits in and corresponding action for that range. Those actions would be turn up the gain if the value is below 210, maintain gain from 210-230, and reduce gain when above 231. To complete these functions it takes approximately 10 cycles every ADC value and 30 cycles once every 2 seconds.

In order to calculate the free cycles the operating parameters of the PIC need to be specified. The PIC has been designed to operate at sub-threshold voltages. In simulations it has been shown to operate properly over a voltage range of 275mV to 350mV at speeds of 400 KHz to 800 KHz. For a baseline we chose to run at 800 KHz with a voltage of 300mV. Even though the input clock frequency is 800 KHz the PIC only gets 200 instructions per sample. This is due to an internal clock divider which provides time for processing interrupts, loading instructions, and other system functions. This results in the worst case being a total of 170 i-cycles for data compression. Since the compression algorithms will fit into 170 i-cycles there is no need to modify the system level architecture.

TABLE I
PICCPU POWER USAGE

	Without Compression	With Compression
NOP Instantaneous Power	318.1nW	318.1nW
AVG Instantaneous Power	322.95nW	324.46nW
Free i-cycles	170	140

B. PIC Power Results

In order to get final metrics on the system we are comparing power usage while completing standard system functionality to that with compression added. This information has been gathered by running an Ultrasim simulation at the schematic level with a detail level of MS (mixed signal). Running a simulation at any higher levels will result in incorrect simulations due to the sub-threshold operation. Using current and power results from Ultrasim do correlate closely to those of a Spectre simulation. From a test simulation it does confirm the 3 percent or less target error which is specified by the Ultrasim manual for this mode. Below figure (12) shows the results of energy consumption per instruction.

When command is a zero then the PIC processor is doing a NOP operation, otherwise it is completed one of its other instructions. As seen in figure (12) there is little variation in the energy between completing a NOP and any other instructions. This data suggest that instead of consuming more energy for doing compression we may end up recovering lost energy due to NOP loops. Seen below in Table (I) displays the power results for running the PIC with and without compression enabled.

From the data above it is easy to see that the processor still spends a majority of it's time in NOP's. Hence the average instantaneous power, over 2ms time, is dominated by that

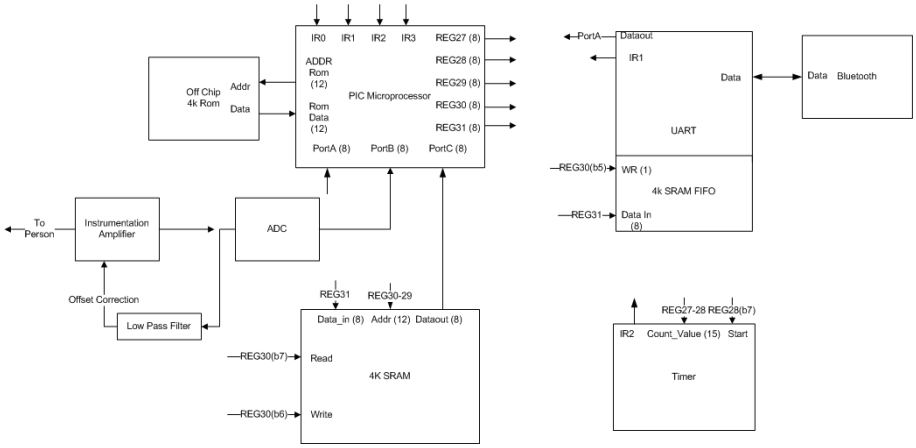

Fig. 11. PICCPU full system diagram.

of the NOP instruction. Even after looking at several other instructions most of them have power usages only 10's of nanowatts above the NOP instruction. As a result the PIC processor draws approximately the same amount of power no matter what it is doing. This information confirms the initial suggestion that we are recovering wasted energy by doing compression.

IV. RF Channel

A. Mixer Design

One of the main components in the RF channel is the mixer. The purpose of the mixer is to act like a multiplier of two RF frequencies. Equation (4) shows the intended functionality of the mixer. The two components remaining are a higher frequency component and a lower frequency component equal to $w_{hf} = w_{LO} + w_{Sig}$ and $w_{lf} = w_{LO} - w_{Sig}$, respectively. In a transmit and receive design you would optimize your mixer to multiply up or down your signal depending on each mode of operation.

$$cos(wt) = \frac{1}{2}(e^{jwt} + e^{-jwt}) \quad (1)$$

$$A_{LO}\cos(w_{LO}t)A_{Sig}\cos(w_{Sig}t) = \frac{A_{LO}A_{Sig}}{4}\left(e^{-jw_{LO}t} + e^{jw_{LO}t}\right)\left(e^{-jw_{Sig}t} + e^{jw_{Sig}t}\right) \quad (2)$$

$$= \frac{A_{LO}A_{Sig}}{4}(e^{-jt(w_{LO}+w_{Sig})} + e^{jt(w_{LO}+w_{Sig})} + e^{-jt(w_{LO}-w_{Sig})} + e^{jt(w_{LO}-w_{Sig})}) \quad (3)$$

$$= \frac{A_{LO}A_{Sig}}{2}\left(\cos(t(w_{LO} - w_{Sig})) + \cos(t(w_{LO} + w_{Sig}))\right) \quad (4)$$

The mixer topology used for this paper is based on the Gilbert Cell. The basic design is shown in Figure (13). Because the Gilbert Cell is an active mixer, it has the additional benefit of amplifying the two signals it mixes; however, the output amplification will eventually roll off. This can be useful as a low pass filter if you are taking two high frequency signals and trying to mix them down to a lower frequency, because the higher frequency component will be attenuated greatly. Transistor sizing and current bias are the main design

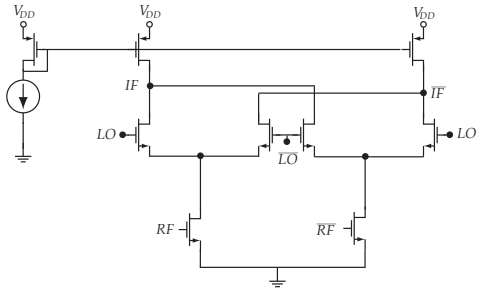

Fig. 13. Gilbert Cell Topology with Active Load Mirror

considerations. The F_{max} for a current bias of 1mA analog common source amplifier with a pmos current mirror load and a pmos and nmos width ratio of $\frac{w_p}{w_n} = \frac{4}{3}$ is about 63 GHz with no load and 24 GHz with the load of a second stage of the same size. The decrease in F_{max} is due to the charging of the gate capacitance on the second stage. Increasing the bias current allows a higher driving ability, another alternative is to decrease the width of the input gate of the subsequent stage; however, this also decreases the driving ability of that stage, since the amount of current driven is decreased in that stage.

The mixers designed for this project lack optimization of parasitic effects. Each design has been tested and measured using Spectre Analog Artist. Some of the effects of wire length between transistor connections are ignored. With more time a more robust design for the 50 GHz mixer could be explored. Given the current design second stage amplifier would be required to buffer the 50 GHz mixer before using a power amplifier. It is recommended to use a buffer amplifier anyway; however, one was not used in the following measurements. To turn off the mixer for "gated-off" power measurements, both the differential RF inputs were set to 0V and leakage power was measured. The power usage was averaged over several RF clock cycles to give a more accurate power consumption, see equation(5).

$$P_{avg} = \frac{1}{T}\int_0^T v(t)i(t)dt \quad (5)$$

$$T = period\ of\ time\ for\ which\ you\ are\ averaging$$

For a local oscillator mixed at 50 GHz with 125 Mbps signal

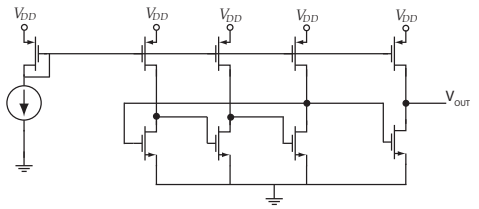

Fig. 14. Ring Oscillator Topology

the power usage of the mixer is $9.23m$W. The gated-off power for this mixer is 33.8μW. A mixer optimized for a lower local oscillator frequency of 2 GHz and a 125 Mbps signal consumes 500μW of power. The gated-off power for this mixer is 2.02μW. A 9-50-GHz mixer was designed in [3], that used a current between $5m$A to $15m$A and a supply voltage of 1.2V. This yields a power consumption of 6-18mW. They used similar transistor technology as the previously designed mixers. More care was taken in active load sizing, and noise figure analysis. No "gated-off" power was given in their paper.

B. Local Oscillator Design

A local oscillator is another necessary component for an RF channel. Two designs were implemented in Cadence, one was actually sent out for fabrication in April. The two designs follow a three stage inverter with the last stage output connecting to the first stage's input terminal. This causes instability and thus a consistent oscillation. See Figure(14). The source terminals for each of the nmos devices are connected to an additional nmos device used as a current throttle. By setting the gate voltage to 0V you can set them in "gated-off" mode.

The first design is similar to the figure 14 except it does not use the pmos devices as current mirrors, it sets the gates to same net as the input to their respective inverter. The F_{max} for this design roughly 1.575 GHz. There are four buffer stages cascaded after the oscillator to drive an on-chip antenna. The average "on" power for the oscillator and buffers is $3.502m$W. The gated-off power is 1.804μW.

The second design is like figure 14. This design can operate at 15GHz. This is partly due to the higher current draw though the pmos devices, as well as the smaller capacitance seen at subsequent inverter stages. The "on" power for this topology is $6.05m$W, with a gated-off power of 69.4μW. The gated off power is much higher for this device because the nmos gate used for throttling, is much larger. The design by sizing it to be larger we decrease the on resistance; however, this also increases the leakage current.

A 51 GHz voltage controlled oscillator (VCO) with similar transistor technology was designed in [4] by Tiebout et al. The core current bias was set from 1-3mA and the supply voltage was set at 1V. This yielded a power consumption from 1-3mW. No off region was specified for the VCO.

C. Amplifier Design

A 60-GHz amplifier was designed by DOAN et al. in [5]. Their technology is similar to the ST120 process we used. The power dissipation for their design was around $36m$A at 1.5V, thus yielding a power consumption of $54m$W. This is because they have three cascaded common source amplifiers with a current bias of $12m$A each. Because of the time restrictions of the course a similar amplifier was not design. We do not have any information on any sleep power operations, for this paper.

V. Conclusion

Thus far, three approaches to power minimization in an electrocardiogram monitor (and by abstraction, an arbitrary BAN system) have been presented. Each provides a degree efficiency in and of themselves, but it is when all elements are merged into a unified system that the global benefits truly take shape.

A. Energy Usage Analysis

One of the premises of this project is to prove that shortening the transmit time will decrease the total power consumption. We can use equation (11) to plot the energy usage for both bits transmitted per sample and transmit frequency. We used the datasheet [6] for figure 15. We left out the energy required to re-acquire communication between the two devices in this graph. We also normalized the energy to one sample in the figure. Since the PICCPU receives 8 bits per sample, the bits are buffered over several samples before transmitting. It is in sleep mode during that time, however the PICCPU continues to run. This figure shows us that as Frequency increases the energy gets closer to $(P_{Sleep} + P_{PIC})\, T_{Sample} + \frac{E_{Acquire}}{N_{Sample}}$, see eqn (13). You can see this relationship as you take the limit of equation (11) as $F_{Tx} \rightarrow \infty$.

$$E_{Sample} = (P_{Sleep}T_{Sleep} + P_{Tx}T_{Tx} + P_{PIC}N_{Sample}T_{Sample} + E_{Acquire})/N_{Sample} \quad (6)$$

$$T_{Sample} = \frac{1}{F_{Sample}} \quad (7)$$

$$N_{Sample} = \frac{N_{bits}}{BitsPerSample} \quad (8)$$

$$T_{Tx} = \frac{N_{bits}}{F_{Tx}} \quad (9)$$

$$T_{Sleep} = N_{Sample}T_{Sample} - T_{Tx} \quad (10)$$

$$E_{Sample} = (P_{Sleep}(\frac{N_{Sample}}{F_{Sample}} - \frac{N_{bits}}{F_{Tx}}) + \frac{P_{Tx}N_{bits}}{F_{Tx}} + \frac{P_{PIC}N_{Sample}}{F_{Sample}} + E_{Acquire})/N_{Sample} \quad (11)$$

$$\lim_{F_{Tx}\rightarrow \inf} E_{Sample} = P_{Sleep}\left(\frac{N_{Sample}}{F_{Sample}} - 0\right) + 0 + \frac{P_{PIC}N_{Sample}}{F_{Sample}} + E_{Acquire})/N_{Sample} \quad (12)$$

$$\lim_{F_{Tx}\rightarrow \inf} E_{Sample} = \frac{P_{Sleep} + P_{PIC}}{F_{Sample}} + \frac{E_{Acquire}}{N_{Sample}} \quad (13)$$

If we take into account compression ratio (14) then we can use equation (15) for T_{Tx} and equation (16) for T_{Sample}. Equation (17) show the final equation using compression. Figure 16 shows energy normalized per sample as the compression ratio changes from 1 to 10. If you look at the compression ratio of 1 in figure 16 you will notice that its slope along frequency matches the slope of figure 15

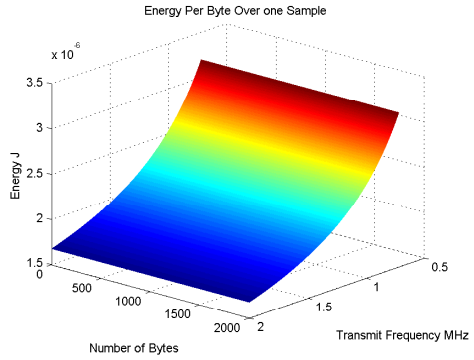

Fig. 15. Plot of energy used during one sample as number of bits changes

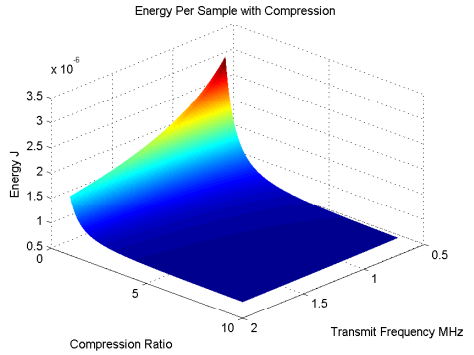

Fig. 16. Plot of energy used during one sample as compression changes

for a 2000Kbytes transmitted.

$$C = Compression\ Ratio \tag{14}$$

$$T_{Tx} = \frac{N_{bits}}{F_{Tx}C} \tag{15}$$

$$T_{Sample} = \frac{C}{F_{Sample}} \tag{16}$$

$$E_{Sample} = (P_{Sleep}(\frac{CN_{Sample}}{F_{Sample}} - \frac{N_{bits}}{F_{Tx}C}) + \frac{P_{Tx}N_{bits}}{F_{Tx}C} + \frac{P_{PIC}N_{Sample}C}{F_{Sample}} + E_{Acquire})/(N_{Sample}C) \tag{17}$$

You can also see a much more drastic energy savings along compression curve. This shows that using compression reaches the asymptotic power level faster than increasing your data rate alone. The drawback to using compressions and having a slow sampling data rate, $8Kbps$, is evident if you are working in a real-time environment. In order to have large the largest savings(C=10 and $N_{BytesTx}$=2KBytes) you would have to wait roughly $\frac{20K\ samples}{1KHz} = 20seconds$ between updates.

B. Future Work

Knowing that the PIC has so many free cycles it would be beneficial to reduce how long it sits in the NOP wait state. This could be done by tailoring the clock speed to allow only enough time to complete the required operations. Since doing this for every program is tedious an auto adjustment feature could be ran during startup so the processor itself can control its clock frequency. The PIC processor now has this ability since it has a free interrupt and additional output ports.

TABLE II
ACTIVE POWER USAGE

	Project	Bluetooth	Citations
Mixer@50GHz	9.23mW	-	18mW
@2GHz	500μW	-	-
LO@2GHz	3.5mW	-	-
@15GHz	6mW	-	-
@50GHz	-	-	1-3mW
Amplifier@50GHz	-	-	54mW
Active Tx Power*	58mW	65mW	-
@ 2GHz			
Active Tx Power*	67.23mW	-	75mW
@ >>2GHz			

* The power total of [5] is used for non-Bluetooth sums

TABLE III
SLEEP POWER USAGE

	Project	Bluetooth
Mixer@2GHz	2.02μW	-
LO@2GHz	1.804μW	-
Sleep Power*	54mW	825μW
@ 2GHz		
Idle Power*	54mW	8.25mW
@ 2GHz		

* The power total of [5] is used for non-Bluetooth sums

Another effect of this realization of the number of free i-cycles is the possibility of implementing more advanced compression algorithms or tailoring the compression to the instantaneous data set in real time. In this way, more effective compression ratios may be obtainable with similar or better lossey characteristics; all with to appreciable impact on performance.

Ultimately, the system described herein illustrates the practicality of multifaceted approaches to power minimization within resource constrained environments.

REFERENCES

[1] S. M. Kim, J. W. Chong, B. H. Jung, M. S. Kang, and D. K. Sung, "Energy-aware communication module selection through zigbee paging for ubiquitous wearable computers with multiple radio interfaces," *Wireless Pervasive Computing, 2007. ISWPC '07. 2nd International Symposium on*, pp. –, 5-7 Feb. 2007.

[2] A. Riemens, R. van der Vleuten, P. van der Wolf, G. Jacob, J. van de Waerdt, and J. Janssen, "Transparent embedded compression in systems-on-chip," *Signal Processing Systems Design and Implementation, 2006. SIPS '06. IEEE Workshop on*, pp. 256–261, Oct. 2006.

[3] C.-S. Lin, P.-S. Wu, H.-Y. Chang, and H. Wang, "A 9-50-ghz gilbert-cell down-conversion mixer in 0.13-/spl mu/m cmos technology," *Microwave and Wireless Components Letters, IEEE*, vol. 16, no. 5, pp. 293–295, May 2006. [Online]. Available: http://ieeexplore.ieee.org/iel5/7260/34136/01626265.pdf

[4] M. Tiebout, H.-D. Wohlmuth, and W. Simburger, "A 1v 51ghz fully-integrated vco in 0.12/spl mu/m cmos," *Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International*, vol. 2, pp. 238–239, 2002. [Online]. Available: http://ieeexplore.ieee.org/iel5/7787/21395/00992232.pdf

[5] C. Doan, S. Emami, A. Niknejad, and R. Brodersen, "Millimeter-wave cmos design," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 144–155, Jan. 2005.

[6] *Roving Networks Bluetooth Module RN-41 CLass 1, 2.0 EDR*, Roving Networks, November 07. [Online]. Available: http://www.rovingnetworks.com/documents/RN-41.pdf